SG04/309

PCT/SG2004/000309

REGISTRY OF PATENTS
SINGAPORE

This is to certify that the annexed is a true copy of following application as filed with the Registry.

Date of Filing	:	23 SEP 2003
Application Number	:	200305550-6
Applicant(s) / Proprietor(s) of Patent	:	LOH, Tien Wai
Title of Invention	:	A METHOD FOR ACCESSING AND DISPLAYING DYNAMIC DATA IN WEB APPLICATION

Chig Kam Tack (Mr)
Senior Assistant Registrar
for REGISTRAR OF PATENTS
SINGAPORE

01 Oct 2004

PCT/SG 2004/ 000309

PATENTS FORM 1
Patents Act
(Cap. 221)
Patents Rules
Rule 19

INTELLECTUAL PROPERTY OFFICE OF SINGAPORE

REQUEST FOR THE GRANT OF A PATENT UNDER SECTION 25

101101

* denotes mandatory fields

Ldgmt Date: 23 Sep 03

1. YOUR REFERENCE* P0001

2. TITLE OF INVENTION* Method For Accessing And Displaying Dynamic Data In Web Applications

3. DETAILS OF APPLICANT(S)* (see note 3)

Number of applicant(s) 1

(A) Name LOH TIEN WAI

Address Blk 4, Sago Lane #04-117

State

Country SG

☐ For corporate applicant

☒ For individual applicant

State of incorporation

State of residency

Country of incorporation

Country of residency SG

☐ For others (please specify in the box provided below)

R78702

R160452/50

23/9

(B) Name

Address

State

Country

ACTION

□ For corporate applicant

State of incorporation □

Country of incorporation □

□ For individual applicant

State of residency □

Country of residency □

□ For others (please specify in the box provided below)

□

(C) Name □

Address □

State □

Country □

□ For corporate applicant

State of incorporation □

Country of incorporation □

□ For individual applicant

State of residency □

Country of residency □

□ For others (please specify in the box provided below)

□

□ Further applicants are to be indicated on continuation sheet 1

4. DECLARATION OF PRIORITY (see note 5)

A Country/country designated □

File number □

Filing Date DD MM YYYY □

B Country/country designated □

File number □

Filing Date DD MM YYYY □

□ Further details are to be indicated on continuation sheet 6

5. INVENTOR(S)* (see note 6)

A The applicant(s) is/~~are~~ the sole/~~joint~~ Inventor(s) Yes ☒ No □

B A statement on Patents Form 8 is/will be furnished Yes □ No ☒

6. CLAIMING AN EARLIER FILING DATE UNDER (see note 7)

□ section 20(3) □ section 26(6) □ section 47(4)

Patent application number □

DD MM YYYY

Filing Date □

Please mark with a cross in the relevant checkbox provided below
(Note Only one checkbox may be crossed)

□ Proceedings under rule 27(1)(a)

DD MM YYYY

Date on which the earlier application was amended □

□ Proceedings under rule 27(1)(b)

7. SECTION 14(4)(C) REQUIREMENTS (see note 8)

Invention has been displayed at an international exhibition Yes □ No ☒

8. SECTION 114 REQUIREMENTS (see note 9)

The invention relates to and/or used a micro-organism deposited for the purposes of disclosure in accordance with section 114 with a depository authority under the Budapest Treaty

Yes □ No ☒

9. CHECKLIST*

(A) The application consists of the following number of sheets

i	Request	0	Sheets
ii	Description	8	Sheets
iii	Claim(s)	2	Sheets
iv	Drawing(s)	2	Sheets
v	Abstract (Note· The figure of the drawing, if any, should accompany the abstract)	1	Sheets
Total number of sheets		13	Sheets

(B) The application as filed is accompanied by·

□ Priority document(s) □ Translation of priority document(s)

☐ Statement of inventorship & right to grant

☐ International exhibition certificate

10. DETAILS OF AGENT (see notes 10, 11 and 12)

Name

Firm

11. ADDRESS FOR SERVICE IN SINGAPORE* (see note 10)

Block/Hse No 4

Level No 4

Unit No /PO Box 117

Street Name Sago Lane

Building Name

Postal Code 050004

12. NAME, SIGNATURE AND DECLARATION (WHERE APPROPRIATE) OF APPLICANT OR AGENT* (see note 12)
(Note: Please cross the box below where appropriate.)

☒ I, the undersigned, do hereby declare that I have been duly authorised to act as representative, for the purposes of this application, on behalf of the applicant(s) named in paragraph 3 herein

Name and Signature

DD MM YYYY
23 1 2003

Description

Field of the Invention

This invention generally relates to the area of dynamic content generation for web pages.

Background of the Invention

Web application development that is currently in existence usually comprises the design of web pages, and the design of programming logic. As each requires different skill sets, it is best that they be handled by two different groups of people, i.e. page designers for designing web pages, and software developers for designing the programming logic.

One pattern that is used to achieve the separation of web page design from the programming logic design is the Model-View-Controller (MVC) pattern as shown in Figure 1. In this pattern, a controller component interprets a request submited by the browser and calls the model components to make the necessary changes. After the necessary changes have been made, the controller then selects and calls the appropriate view component to generate a response back to the browser.

Technologies like JavaServer Pages (JSP) and Microsoft Active Server Pages (ASP) have been popular for developing web applications that provide dynamic content. They allow page designers to mix HTML code with scripting code and xml-like tags capable of encapsulating programming logic, to generate dynamic content web pages. This capability is often used for implementing the view component of the MVC pattern. A typical implementation of MVC pattern using Java Servlet and JSP technologies is shown in Figure 2, wherein the controller components are implemented using java servlets, the model components using java objects, and the view components using JSP pages.

Page designers using these existing technologies can design and change a web page (the view component) without altering or understanding the programming logic (the controller and model components). However, many a times, due to the complexity of the web pages, the page designers must still have working knowledge of the scripting language to design a web page of any complexity. One common solution is to use tags capable of encapsulating programming logic to eliminate the use of scripting language in web pages. However, indiscrimate use of tags to encapsulate the programming code leads to the proliferation of tags. This is further aggravated by the need to represent and access data of different types of data sources that usually resulted in creating a new set of tags for each type of data sources. This approach will also lead to requiring page designers to understand the different domain area of which these tags encapsulate in order to use them correctly. One example will be a tag that encapsulates the code for accessing data in a database via JDBC but requires the page designers to correctly specify the JDBC driver and the database URL (Uniform Resource Locator) information in the tag attributes. The required understanding of other domains, in this case, the possible values of the JDBC driver and the possible values of the required database URL, by the page designers greatly undermines the goal of role separation of page designers and software developers.

It is therefore desirable to have a method that is able to reduce or even eliminate the use of programming language in web pages. Such a method would use only a manageable number of tags and it would not require the user to have prior knowledge of other domains.

Summary of the Invention

The invention identifies a set of use-patterns of dynamic data in web pages. It also defines a set of interfaces for accessing dynamic data of diverse data sources based on the aforesaid use-patterns. This will ensure that mismatches between how dynamic data is being used in web pages and how dynamic data is being accessed, are eliminated. The result is the reduction or even elimination of the need of scripting codes often used for bridging such mismatches in web pages.

Further, a method is provided for accessing data using a set of tags in web pages. This facilitates the use of the aforementioned set of interfaces.

Description of the Drawings

Figure 1 illustrates a sample model-view-controller pattern.

Figure 2 illustrates a sample model-view-controller pattern using Java Servlet and JavaServer Page technologies.

Figure 3 depicts the relationship between the components found in the invention.

Detailed Description of the Invention

Methods consistent with the invention facilitate access and display of dynamic data in web pages. This includes a set of use-patterns, a set of interfaces, and a set of tags. These are now described.

A Set Of Use-Patterns

A set of patterns of the use of dynamic data content in web pages is defined. The set of patterns includes:

Pattern 1. Accessing and displaying the string value of a data item.

Pattern 2. Iterating through a collection of data objects.

Pattern 3. Determining whether specific data items in a collection of data objects contain a specific value.

A data object consists of data items with values. An example of a data object is an employee record with data items first_name, last_name, gender and age whose values are "Smith", "John", "Male" and "35" respectively.

The aforementioned set of patterns hereinafter will be referred to as "MyDataUsePatterns".

A Set Of Intefaces For Accessing Data Of Diverse Data Sources

A set of interfaces for accessing data of diverse data sources supporting the MyDataUsePatterns is provided. The set of interfaces comprises:

Interface 1. An interface for accessing the string value of a data item. This interface supports Pattern 1 of MyDataUsePatterns. An example will be an interface that contains the following method:

```
// Return the string value of the specified item.
String getValue(String item)
```

Interface 2. An interface for iterating through a collection and getting the size of a collection. This interface supports Pattern 2 of MyDataUsePatterns. An example will be an interface that contains the following methods:

```
// Move to before the first element.
void beforeFirst()

// Return true if there is next element else return false.
boolean hasNext()

// Move to the next element. Return true if there is next element else return
// false.
boolean next()

// Return the size of the collection. If the size could not be determined
// return -1.
int getSize()
```

Interface 3. An interface for determining whether specific data items in a collection of data objects contain a specific value. This interface supports Pattern 3 of MyDataUsePatterns. An example will be an interface that contains the following method:

```
// Return true if the values of the specified data item in a collection
// contain the specified match value.
boolean containsValue(String item, String matchValue)
```

As this is a set of interfaces, it can be used to access data of different type of data sources by using an appropriate adapter object. Details regarding the use of an adapter to convert the interface of a class into another interface a client expects can be found in most design pattern literature. One recommention is the book entitled "Design Patterns" by Gamma, Helm, Johnson & Vlissides, ISBN 0-201-63361-2.

The aforementioned set of interfaces hereinafter will be referred to as "MyDataAccessInterfaces".

Object that implements one or more of the interfaces of MyDataAccessInterfaces hereinafter will be referred to as "MyDataAccessObject".

A Set Of Tags

A set of tags facilitating the use of MyDataAccessInterfaces in web pages is defined. This includes but not restricted to tags that perform one or more of the following functions:

displaying the value of a data item,

iterating through a collection of data objects,

getting the size of a collection,

evaluating the tag body based on the result of testing the value of a data item,

evaluating the tag body based on the result of testing the size of a collection, or

evaluating the tag body based on the result of testing the values of a data item in a collection of data objects.

The aforementioned set of tags hereinafter will be referred to as "MyDataTags".

A description of some sample tags is as follows:

Tag: data: The data tag obtains and displays the string value of a data item. For example, the tag "mydatatag:data name="employee" item="first_name"" will retreive and display the data item "first_name" of the MyDataAccessObject named "employee". The MyDataAccessObject in this case shall at least implement Interface 1 of MyDataAccessInterfaces.

Tag: iterator: The iterator tag iterates over a collection of data elements, and is used in conjunction with the data tag. For example, the tag "mydatatag:iterator name="employees"" with tag body that includes the data tag "mydatatag:data name="employees" item="first_name"" will iterate through all employees and display their first name. The MyDataAccessObject in this case shall at least implement Interface 1 and 2 of MyDataAccessInterfaces.

Tag: iteratorSize: The iteratorSize tag displays the size value of a collection. For example, the tag "mydatatag:iteratorSize name="employees"" will get and display the size of a collection of employees. The MyDataAccessObject in this case shall at least implement Interface 2 of MyDataAccessInterfaces.

Tag: dataIn: The dataIn tag tests if specific data items of a collection of data objects contain a specific value. If the test returns true, then the body of the tag is evaluated. For example, the tag "mydatatag:dataIn name="employees" item="first_name" match="Smith"" would determine if the first_name of one or more employees are "Smith". If the test returns true, the body of the tag will be evaluated. The MyDataAccessObject in this case shall at least implement Interface 3 of MyDataAccessInterfaces.

Tag: dataNotIn : Contrary to the dataIn tag, the dataNotIn tag tests if specific data items of a collection of data objects do not contain a specific value. If the test returns true, the body of the tag is evaluated. The MyDataAccessObject in this case shall at least implement Interface 3 of MyDataAccessInterfaces.

Tag: dataEqual: The dataEqual tag tests if the value of a data item equals to a specific value. If the test returns true, then the body of the tag is evaluated. For example, the tag "mydatatag: dataEqual name="employee" item="first_name" match="Smith"" would determine if the data item "first_name" of MyDataAccessObject named "employee" is equal to "Smith". If the test returns true, the body of the tag will be evaluated. The MyDataAccessObject in this case shall at least implement Interface 1 of MyDataAccessInterfaces.

Tag: dataNotEqual: Contrary to the dataEqual tag, the dataNotEqual tag tests if the value of a data item is not equal to a specific value. If the test returns true, then the body of the tag is evaluated. The MyDataAccessObject in this case shall at least implement Interface 1 of MyDataAccessInterfaces.

Tag: dataItemEqual: The dataItemEqual tag tests if the values of two data items are equal. If the test returns true, then the body of the tag is evaluated. For example, the tag "mydatatag: dataItemEqual name="employee" item="first_name" matchname="manager" matchitem="first_name"" would determine if the data item "first_name" of MyDataAccessObject named "employee" is equal to the data item "first_name" of another MyDataAccessObject named "manager". If the test returns true, the body of the tag will be evaluated. The MyDataAccessObjects in this case shall at least implement Interface 1 of MyDataAccessInterfaces.

Tag: dataItemNotEqual : Contrary to the dataItemEqual tag, the dataItemNotEqual tag tests if the valuesof two data items are not equal. If the test returns true, then the body of the tag is evaluated. The MyDataAccessObjects in this case shall at least implement Interface 1 of MyDataAccessInterfaces.

Tag: dataItemIn: The dataItemIn tag tests if specific data items of a collection of data objects contain the value of a specific data item of another data object. If the test returns true, then the body of the tag is evaluated. For example, the tag "mydatatag:dataItemIn name="employees" item="first_name" matchName="manager" matchItem="first_name"" would determine if the first_name of one or more employees equal to the first_name of the manager. If the test returns true, the body of the tag will be evaluated. The MyDataAccessObject "employees" in this case shall at least implement Interface 3 of MyDataAccessInterfaces. The MyDataAccessObject "manager" in this case shall at least implement Interface 1 of MyDataAccessInterfaces.

Tag: dataItemNotIn: Contrary to the dataItemIn tag, the dataItemNotIn tag tests if specific data items of a collection do not contain the value of a specific data item of another data object. If the test returns true, then the body of the tag is evaluated. The MyDataAccessObject for the collection shall at least implement Interface 3 of MyDataAccessInterfaces. The MyDataAccessObject for data item shall at least implement Interface 1 of MyDataAccessInterfaces.

Tag: iteratorSizeEqual: The iteratorSizeEqual tag tests if the size of a specific collection equals to a specific value. If the test returns true, then the body of the tag is evaluated. For example, the tag "mydatatag: iteratorSizeEqual name="employees" match="0"" would determine if the size of the collection of employees equals to 0. If the test returns true, the body of the tag will be evaluated. The MyDataAccessObject in this case shall at least implement Interface 2 of MyDataAccessInterfaces.

Tag: iteratorSizeNotEqual: Contrary to the iteratorSizeEqual tag, the iteratorSizeNotEqual tag tests if the size of a specific collection not equals to a specific value. If the test returns true, then the body of the tag is evaluated. The MyDataAccessObject in this case shall at least implement Interface 2 of MyDataAccessInterfaces.

Figure 3 depicts the relationship between the components in the invention. MyDataUsePatterns is defined according to the use-patterns of dynamic content in web pages. MyDataAccessInterfaces is defined to support MyDataUsePatterns. MyDataTags is created to facilitate the use of MyDataAccessInterfaces in web pages. Technologies that support page generation by mixing html code and tags capable of executing programming logic will need to incorporate or support MyDataTags. By incorporating or supporting MyDataTags, these technologies can then use MyDataTags to access domain data, represented in the form of MyDataAccessObject, in their web page.

MyDataAccessObjects need to implement one or more interfaces of MyDataAccessInterfaces to be able to be accessed by MyDataTags.

An Example

To illustrate how the invention works, the JSP code of a web page showing an employee record is shown below.

ViewEmployee.jsp

```
<%@ taglib uri="/src/tags/taglib.tld" prefix="mydatatag" %>
<html>

<head>
   <title>View Employee</title>
</head>

<body>

<b>First Name:</b><mydatatag :data name="employee" item="first_name"/><br>
<b>Last Name:</b><mydatatag :data name="employee" item="last_name"/><br>
<b>Gender:</b><mydatatag :data name="employee" item="gender"/><br>
<b>Age:</b><mydatatag :data name="employee" item="age"/><br>
<hr>
<b>Work History</b><br>
<table>
   <tr>
          <td>From Year</td>
          <td>To Year</td>
          <td>Company</td>
          <td>Position</td>
   </tr>
   <mydatatag :iterator name="work_history">
   <tr>
          <td><mydatatag :data name="work_history" item="from_year"/></td>
          <td><mydatatag :data name="work_history" item="to_year"/></td>
          <td><mydatatag :data name="work_history" item="company"/></td>
          <td><mydatatag :data name="work_history" item="position"/></td>
   </tr>
   </mydatatag:iterator >
   <mydatatag:iteratorSizeEqual name="work_history" match="0">
   <tr>
          <td colspan=4>No record found!</td>
   </tr>
   </mydatatag:iteratorSizeEqual>
```

```
</table>

</body>
</html>
```

The above code uses two MyDataAccessObjects, one for accessing the employee data and one for accessing the work history of the employee. The MyDataAccessObject for the employee data has implemented Interface 1 of MyDataAccessInterfaces, and contains data items first_name, last_name, gender and age. The values of these data items are accessed and displayed using the data tag from MyDataTags. The MyDataAccessObject for the work history has implemented Interface 1 and 2 of MyDataAccessInterfaces, and the collection of work history data with data items from_year, to_year, company and position. The iterator tag from MyDataTags is used for iterating over the collection of work history data and the data tag from MyDataTags is used for accessing and displaying the data items of work history data. The iteratorSizeEqual tag from MyDataTags is used for testing the size of the collection of work history and if it is equals to 0, display the text "No record found!".

The foregoing descriptions of specific embodiments of the invention have been presented for purposes of description and illustration. It should not limit the invention to the precise forms disclosed. It is intended that the specification and examples be considered as sample information only. The full scope of the invention shall be defined by the appended claims.

Claims

1. A method for accessing and displaying dynamic content in web pages, comprising:

   defining a set of use-patterns of dynamic content in web pages;

   defining a set of interfaces for accessing data of diverse data sources that match the set of use-patterns of dynamic content in web pages; and

   defining a set of tags, wherein the tags facilitate the use of the set of interfaces to eliminate programming code in dynamic content generation;

2. The method of claim 1, wherein the set of use-patterns comprise patterns for:

   accessing and displaying the string value of a data item;

   iterating through a collection of data objects; and

   determining whether the data items in a collection of data objects contain a specific value,

   where a data object consists of data items with values.

3. The method of claim 1, wherein the set of interfaces includes:

   interface for accessing the string value of a data item;

   interface for iterating through and getting the size of a collection of data objects; and

   interface for determining whether the data items in a collection contain a specific value,

   where a data object consists of data items with values.

4. The method of claim 1, wherein the set of tags faciliating the use of the set of interfaces in claim 3 to access data of diverse data sources in dynamic content generation..

5. The method of claim 1, wherein the set of tags includes but not restricted to tags that perform one or more of the following functions: displaying the value of a data item, iterating through a collection of data objects, getting the size of a collection, evaluting tag body based on the result of testing the value of a data item, evaluating the tag body

based on the result of testing the size of a collection, or evaluting tag body based on the result of testing the values of a data item in a collection of data objects.

6. The method of claim 1, 4 and 5, wherein the set of tags is used with a platform capable of generating web pages by mixing html code with tags that encapsulate programming logic to generate dynamic web pages.

163163

G00001

Drawings

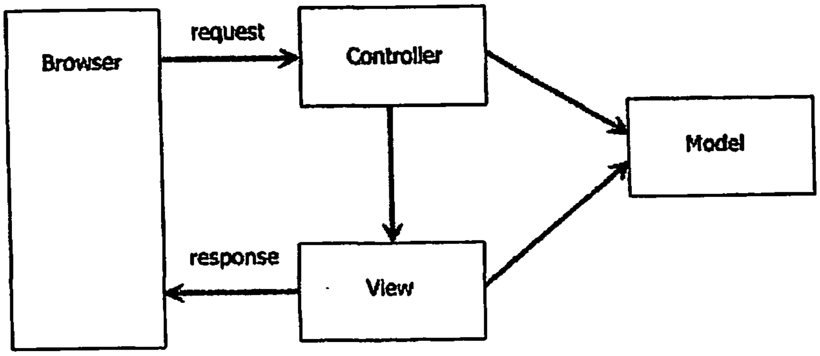

Figure 1

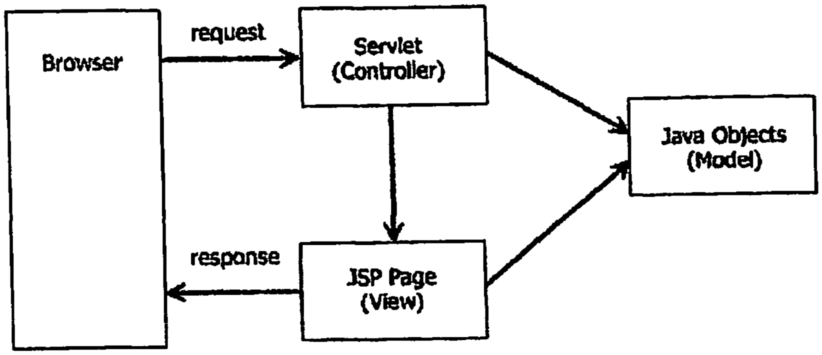

Figure 2

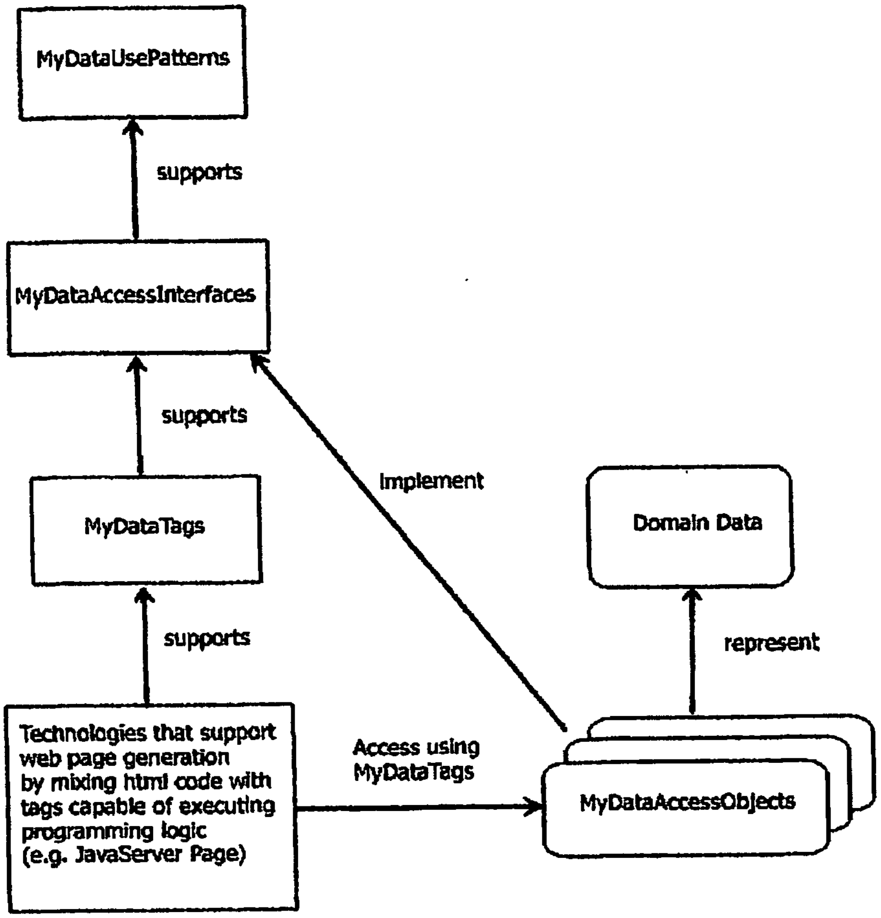
MyDataUsePatterns
supports
MyDataAccessInterfaces
supports
Implement
MyDataTags
Domain Data
supports
represent
Technologies that support web page generation by mixing html code with tags capable of executing programming logic (e.g. JavaServer Page)
Access using MyDataTags
MyDataAccessObjects

Figure 3

Abstract

This patent is about a method for accessing and displaying dynamic data in web applications. These data are from diverse data sources. The method includes a platform that supports web page generation by mixing HTML (HyperText Markup Language) with xml-like tags capable of encapsulating programming logic, a set of interfaces, and a set of tags that facilitates the use of the aforesaid set of interfaces. The benefit of using such a method is that it offers simpler presentation code, and better separation of programming logic from page design.